IN THE SPECIFICATION:

On page 26, please amend paragraph [0098] as follows:

--Figure 4(a) is a screenshot detailing how a user defines a template. In a navigation area 402, the user can access a file organizational structure. This structure can be a file tree, with folder and file organization similar to that found on systems such as popular operating systems ~~used on both~~ Microsoft® Windows® and Apple® ~~Macintosh~~ Mac OS® ~~systems~~. Other file organizational structures can also be used. With the navigation area 402, the user can navigate through the web pages and templates stored in the web publishing system 100, choose to edit existing pages and templates or create new pages and templates, and decide where within the organizational structure to store such templates and pages. A command area 404 provides additional functions available to the user.--

On pages 27 through 28, please amend paragraph [0102] as follows:

--Figure 4(b) is a screenshot showing a template preview presented to the user after the user clicks on the preview button 410 of Figure 4(a). After the user clicks on the preview button 410, the client interface module 124 sends a preview request to the communication module 134, which passes the request to the preview module 130. The preview module 130 then sends the request to the web page module 114. The web page module retrieves the template file from the template module 118. The web page module [[118]] 114 then sends the template file to the communication module 134, which sends the template file to the client interface module 124. The client interface module 124 then displays the template. As the user is in the design UI when previewing a template, the template is displayed nearly the way it would appear as part of a web page generated from the template and a content file. However, instead of being combined with a

content file to form the web page, default content 412 is used. The default content is typically created as part of the template file. The default content takes the place of user-entered content that would normally appear in a web page generated from combining the template and a content file. This allows the user to preview the template without having to also generate a sample content file, and shows where the content from the content file would go in web pages based on the template.--

On pages 31 through 32, please amend paragraph [0110] as follows:

--Figure 4(e) is a screenshot showing how a user enters content to create a content file for a web page after selecting the page edit symbol 422. The content entering screen includes the navigation area 402 and command area 404, as well as a content entering area 424. Information [[427]] on the content entering screen also shows the identity and location within the file structure of the page for which the content is entered. The user may define where in the file structure the page based on the content will be stored, or may define where all pages based on a certain template will be stored. In different embodiments, this can be done when the template associated with the content files is created, when the content file is created, or at other times through use of the file structure manager module 115 to govern the overall file structure. ~~When entering content to create a content file for a web page, a user needs little or no knowledge of HTML~~ A user needs little or no knowledge of HTML to enter content to create a content file for a web page. In the example of Figure 4(e), the user simply types the information into labeled fields. The information entered in this page forms the content file of a web page, as opposed to the template information that is common to each page based on that template. For example, in the field labeled[[,]] "Member_Name" 426, the user types in [[the]] a name. As seen in Figure 4(e), this is a straightforward operation. After typing in all the content, the user clicks on the

"Save" button 428. This causes the client interface module 124 to send the content information that the user entered to the communication module 134. The communication module 134 sends the content information, in the form of a content file, to the content update module 128. The content update module 128 then sends the content file to the content module 120, where the content file is stored in the database 142. The user can also delete or edit the content on this screen, and save the revised content. Thus, creation and editing of the pages is a very simple task.--

On pages 41-43, please amend paragraphs [0142]-[0144] as follows:

--One feature, found in some embodiments, is the automatic linking of documents and the automatic inclusion of content in documents. In one aspect, a document is automatically linked to other documents. ~~Alternately~~ Alternatively, content from other documents is automatically included in a document. In another aspect, links are automatically created within a single page. ~~Alternately~~ Alternatively, content from within a page is listed or otherwise presented again on that same page. The following description provides a general description of automatic linking. Appendix II provides more detail on how automatic linking is accomplished in one embodiment of the invention.

Linking to Other Pages:

Some embodiments where links to other web pages are automatically generated use the location of the other web pages to accomplish that link generation. Figure 8 is a block diagram showing a simplified overview of an embodiment of a file organizational structure. File organizational structures are well known in the art. In the file organizational structure shown in Figure 8, the general structure is that of folders and files. A first folder 802 (also labeled as

"~~folder~~ Folder 1" in Figure 8) is the top level of the organizational structure. The first folder 802 contains documents 804, 806, 808, and 810 and other folders 812, 814, 816, and 818. The folders 812, 814, 816, and 818 may in turn contain documents, other files, and other folders. This can be seen with the fifth folder 818 (also labeled as "Folder 5" in Figure 8). The fifth folder 818 contains documents 822, 824, 826, and 828, as well as a sixth folder 820 (also labeled as "Folder 6" in Figure 8). The sixth folder 820, in turn, can contain still more documents, files and folders.

The web publishing system 100 organizes the web pages created in a similar organizational structure. Content files can be treated as web pages for organizational purposes, since it is known which template the content files will be merged with to form the web pages. Thus, the content files can be treated as having the same location within the organizational structure as the web pages that will result from the merging of the template with that content file. Thus, the web pages and content files are treated as being within folders in an organizational structure. The user can see where in the organizational structure content files/web pages are stored by using the navigation area 402, shown in Figures 4(a) through 4(f) ~~and other Figures~~. The navigation area 402 shown in Figure 4(a) shows one top-level folder. The user can click on the top-level folder to see the files and folders within that top-level folder. The user can then click on folders within the top-level folder to see the contents within those folders. The file structure manager module 115 stores the organization of the files used in the web publishing system 100.—

On pages 45-46, please amend paragraphs [0152]-[0155] as follows:

--In some cases, no ~~other~~ documents may meet the specification for linked documents. The user may define information or content that is to be displayed in such a case. Then, instead

of having a blank document when no documents meet the specification, the user-defined information or content is displayed.

To remain accurate, the list page 902 should be updated when linked documents meeting the specification[[s]] for documents to be listed on the list page 902 are added or changed. In some embodiments, the list page 902 is updated when the user specifies that the list page 902 should be published. When this occurs, the web page module 114 forms the list page 902 from the template and content files. The web page module 114 reads the link specification[[s]] in the template file for the list page 902 and retrieves the relevant files to be linked and their locations from the file structure manager module 115. The web page module 114 then inserts the links to the linked documents into the web page formed from the template and content files.

In other embodiments, the file structure manager ~~structure~~ module 115 searches all the templates in the system whenever a content file is created or changed. The file structure manager ~~structure~~ module 115 determines whether the new or changed content should be reflected in any list documents 902. If so, the file structure manager ~~structure~~ module 115 sends instructions to remove the current version of the list document 902 from cache, so that the next time the list document 902 is needed, it is formed anew with the correct list of links instead of retrieved from cache. The file structure manager ~~structure~~ module 115 may also immediately cause the web page module 114 to form the list document 902 with the correct list of links and send the list document 902 to the publish module 116, which sends the list document 902 on to the user server 110. ~~Alternately~~ Alternatively, the web page module 114 may wait until the created or changed linked documents are published on the user server 110 via the publish module 116, and automatically publish the new, correct list document 902 at the same time.

Such link pages 902 can also be automatically formed in systems that do not follow the template/content scheme. In such systems, the link page would still include a specification for

which pages to link to. A file structure manager ~~structure~~ module 115 can track when pages meeting the specification are saved in the system, and automatically trigger the updating of the link list in the link page. Thus, when a page is created by a user, and the user selects a "save" command, the page is saved. The file structure manager ~~structure~~ module 115 also automatically checks the files in the system to see if the saved page meets the specification within link pages. If so, the link page is automatically updated. This is particularly useful when linking to non-HTML content such as multimedia files (sound files, movie files, etc.) and text documents (documents in .pdf format, in Microsoft® Word® format, etc.).--